**Escape Room Season 1: System Design Document**

# Table of Contents

[1 Table of Contents i](#_Toc494748867)

[2 Objective 1](#_Toc494748868)

[2.1 Background 1](#_Toc494748869)

[3 System Overview 1](#_Toc494748870)

[4 Programming the System 2](#_Toc494748871)

[4.1 Coding Practices 3](#_Toc494748872)

[4.1.1 Naming Conventions 3](#_Toc494748873)

[4.1.2 Function Management 3](#_Toc494748874)

[5 Setup of System Devices 3](#_Toc494748875)

[5.1 Testing of Device Specifications 3](#_Toc494748876)

[5.2 Testing Placement of Devices 5](#_Toc494748877)

[6 Maintenance 5](#_Toc494748878)

[6.1 Circuit Schematics 5](#_Toc494748879)

[6.2 Devices and Their Placement in Arduino Files 10](#_Toc494748880)

[6.3 Logic behind the Devices: Coding 11](#_Toc494748881)

[6.3.1 NFC Card and El Wire 11](#_Toc494748882)

[6.3.2 Bluetooth Communication 12](#_Toc494748883)

[6.4 Maintenance for Adding/Replacing Devices 12](#_Toc494748884)

[6.4.1 Replacing Parts 12](#_Toc494748885)

[6.4.2 Adding Parts 12](#_Toc494748886)

[6.5 Testing 13](#_Toc494748887)

[7 Wiring and Enclosures 13](#_Toc494748888)

[8 Budget 13](#_Toc494748889)

[9 Conclusion 15](#_Toc494748890)

[10 References 15](#_Toc494748891)

# Objective

This document provides information on the Escape Room Season 1 project, which will be called **Escape Room 1** and **System** in this document. This document was created with long-term storage of the project between uses and maintenance of the System. This document is subject to change due to modifications to the Escape Room 1 project.

NOTE: This document does not discuss puzzle ideas or reasoning behind the final implemented puzzles.

The parts of Escape Room 1 this document discusses include hierarchy of system devices, languages used in programming the system and coding practices thus, setup of system devices, maintenance of the System for general use and when modifying system devices, wiring of the System, and budget of the Escape Room 1 project.

## Background

This project’s main objective was to create an escape room using a large amount of technology not usually found in escape rooms, which is why this section will briefly describe the basic principles and general setup/goal for an escape room.

The basic principle of an escape room is that it is an activity where an individual or a group of individuals are placed within a space which is usually a room within a building or an entire building depending on the escape room scope that tends to be filled with puzzles and a countdown timer. The escape room activity usually ends when the timer runs out of time or when the individual(s) solve all the puzzles within the escape room space.

The general setup of an escape room is a room that is filled with props to setup the story environment, and props to use to solve puzzles located within the room. There tends to be smaller puzzles (**sub-puzzles**) that are subsections of one larger puzzle, which is the overall goal of the escape room. All puzzles are planned out by the escape room designer and are usually designed to take an average person the full time on the timer to solve all of the puzzles. The difficulty also depends on the escape room designer; the previous sentence is the most often setup preferred by escape room designers.

# System Overview

Escape Room 1 deals with a variety of devices that can be viewed in the Budget section of this document. The overall hierarchy structure of the System can be viewed in Figure 1 and begins with a Raspberry Pi 2 model A (a.k.a. Pi) which contains the logic for the System’s puzzles. This primarily deals with both device to device, and puzzle to puzzle transitions. In general terms, the Pi coordinates device interactions within a smaller sub-puzzle in order to manage and complete the sub-puzzle as well as coordinates transitions between sub-puzzles in order to complete the larger puzzle. The Pi tends to do this by sending strings containing command codes via Bluetooth to various micro-controllers. In this project, we use 4x Arduino Mega 2560s rev2 with HC-05 Bluetooth modules which are capable of both master and slave modes as our micro-controller of choice. The Pi also manages a keyboard and a display for output, which can be used for either admin use or for Escape Room 1 puzzles.

As mentioned in the previous paragraph, there are 4 Arduinos used in the System for the primary use of managing devices necessary for the functionality of Escape Room 1. These are generally then grouped into locations where the devices will be used and same devices are grouped together as much as possible. There are 2 Arduinos located within a Box container and for easier reading, the first Arduino will be called the NFC Arduino and the second Arduino will be called the Main Cube Arduino. There are also 2 Arduinos that handle other devices elsewhere in the room and these will be called the Magnet Arduino and the Laser Arduino. These Arduinos manage the majority of devices needed within Escape Room 1, which include but is not limited to lasers, NFC reader, solar cells for sensors, and a keypad. For a more in depth list, please refer to the Budget section.

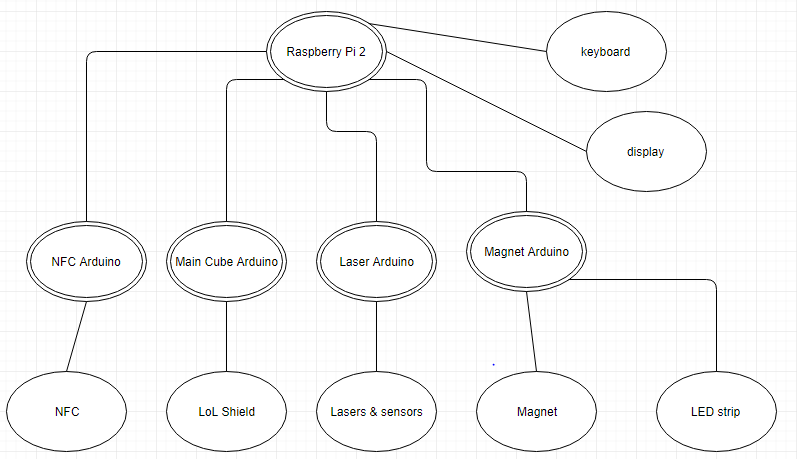


Figure : Device Structure of Escape Room 1.

# Programming the System

There are 2 main programming languages used within the System. Python was used to program the Pi because there was a large pool of python examples provided online by the Raspberry Pi & Python community and it provided easy implementation for various puzzle options that were thought of in the System Design stage, but are not implemented in the final product. The programming in the Python script includes Bluetooth communication between the Pi and the various Arduinos, uni-curses for display using Python, and the logic for sub-puzzle/escape room completion.

To program the Arduinos, the Arduino library and language was used which contains a c++ similar language. The NFC Arduino handles all transmitting CAT5 puzzle pieces, PN532 (Arduino capable NFC reader/writer), 12 LED ring, and the cube inner light. The Main Cube Arduino handles displaying the necessary lit lights on the LoL shield, keypad membrane, and receiving all CAT5 puzzle pieces. The Magnet Arduino handles an electromagnet, LED strip for lighting within the room, and 5 servo motors. The last Arduino, the Laser Arduino, manages the sensors, lasers, small LED diodes, and a PanTilt servo motor. The specific programs can be found in the Appendix of the document or in the Escape Room code repository/google drive.

## Coding Practices

When the use of a symbol is described in this section, the symbol will be shown in parenthesis.

### Naming Conventions

For both the Arduino code and the Python code used for the Raspberry Pi, lowerCamelCase naming convention is used for all variables and function/method names not created by a ‘#define’ method. UpperCamelCase is used for all variables defined by a ‘#define’ method and all classes/objects.

### Function Management

All function blocks of code start with an opening brace ( { ) and end with a closing brace ( } ). In addition, all functions and global variables that correspond to a specific device or functionality can be found in sub sections at the beginning of the Arduino program (.ino) files, where each sub section begins with a C++ multi-line comment which describes the functionality and/or device the sub section is used for and the sub section ends with a C++ multi-line comment which explicitly says it is the end and re-iterates what functionality and/or device the sub section dealt with.

# Setup of System Devices

This section describes the setup of all devices for the System which includes how the placement of devices was done, and what testing was done to various electronic devices in order to find the necessary working conditions which can be found in the finished circuit schematics that can be found in the Maintenance section.

## Testing of Device Specifications

This section describes what devices we verified the working specifications for and what the working specifications are. We had to check what the working specifications were because the majority of the specifications found in datasheets are maximum ratings and actual ratings vary between devices.

We tested the 5mW red lasers which had maximum ratings (maximum working specification) at 5V when working with a current of 25mA but when testing we found that it works at 5V when working at a current of 20mA. We also tested 2 kinds of solar cells to use as sensors for the lasers, where we got the first kind of solar cells from the local Princess Auto store in the Surplus section. These solar cells were too fragile and broke often, and did not have any wires with a solder connection on them to allow for easy wiring. In addition to the second kind of solar cells which were found on the Digikey website, that cost approximately the same amount of money (not including shipping cost) but had solder connections to wires for easy wiring and an online datasheet.

We also tested the NeoPixel ring which was found on the Adafruit website by soldering on wires for the necessary connections and running various example programs which were given with the installation of the Adafruit NeoPixel library. With the use of the library, we found the desired brightness we want the ring to be was 16% brightness (40 out of 255-full brightness) and we found that the type of the ring setup which we bought was NEO\_GRBW which was important because it is needed in the creation of the Adafruit NeoPixel object in the Arduino programming code.

We also tested the **LoL** (Lots of LEDs) **shield** which we borrowed from a Professor at the University of Regina called David Gerhard, by using example programs found online after some research. During this research we found out that the unprotected backs of the soldered connections may touch the exposed metal on the Arduino which will cause the LoL shield to act unpredictably when the Arduino is running the example programs.

The last device we tested to ensure we understood how it works and how to program it was the NFC reader/writer which is compatible with an Arduino. We found the NFC reader on the Adafruit website which is called the **PN532**. Here we tested how far away it could detect various NFC cards and tags, and we found that we can scan the bus passes used by Regina Transit the farthest amongst all the testing NFC cards and tags. We tested basic NFC tags (Mifare Ultralight) which were found on the Amazon website which had to have direct contact with the PN532, we tested the Mifare Classic NFC card which was given with the PN532 and it was detected at a maximum distance of 1cm, and we tested Mifare Classic NFC card which I used over the years as a Regina bus pass (which is now retired as a bus pass) and it had a maximum distance of 4cm. For the testing, all cards were held so the length of the NFC card and the length of the reader were held parallel to each other which is the optimal positioning of the card to get detected. We found this optimal positioning information off the Adafruit NFC information page (L., 2012) and from personal experimentation.

We also tested the Bluetooth communication between devices in order to test that communication between devices was possible for the System. Unfortunately we only tested the use of the Bluetooth communication in the situation where there was constant communication between one Arduino and one Raspberry Pi, which we realised later on in the project that this was not always the expected usage for the communication.

## Testing Placement of Devices

For various devices we used temporary circuits in order to determine the placement we want the devices to be within the room. For the rest of the devices used in the System, we chose locations in the room for the devices where the devices and device enclosures would make the room seem smaller and more esthetically pleasing to the Escape Room planner, Oles Shnurovskyy.

For the placement of the lasers, we connected one laser to one 3V CR2025 watch battery and repeated this as many times as we had loose lasers (lasers not located within an enclosure) to place within the room.

We also tested out the placement of 10mm LED diodes/lights individually by attaching one 3V CR2025 watch battery to them and we tested the 5mm LED diodes/lights individually by attaching a small circuit which ensured a current of 20mA at an assumed voltage of 3V for the diode/light.

# Maintenance

This section deals with the maintenance of the System, which includes how the System was first created, and what steps should be made in order to do changes easily. In order to ensure ease of maintainability, this section contains circuit schematics of the System which are each divided into sections based on electronic device. This section also describes what devices/functionalities each Arduino within the System controls/manages and describes the logic of the programming code that corresponds to each device/functionality.

In addition, this section also briefly describes the programming code meant for testing each device/functionality in order to ensure ease of testing or modification of tests after a device or functionality is modified.

NOTE: You can find the programming code that corresponds to each device at the top of the associated Arduino programming (.ino) file.

## Circuit Schematics

This section contains schematics for all the electronic circuits within the System. At the time this document was created, the circuit diagrams were incomplete due to this document containing only part 1 of the Escape Room Season 1 system. There may be more schematics added for additional escape room activities/parts.

Schematic Background

Some basic schematic information is included in this section. If the reader is knowledgeable in reading schematics, please ignore this section.

Basic circuits require a power supply and a ground in a circuit in order to do work. The power supplies for the circuits in the section are either denoted as a ‘pin IN’ for the various devices or are labelled as a specific power supply voltage. If the schematic shows a specific power supply, then the power supply is either a wall charger specifically meant for the System or is an equivalent voltage supply by use of batteries in series. The ground is where all the current flows towards, and in this section the symbol  means that a specific wire is connected to the ground pin located on the associated Arduino and/or Raspberry Pi; whereas if we use the negative contact of a battery for a circuit, then you will see an explicit label in the schematic saying what the voltage supply value is and then it says it is a ground. A good example of this is Figure 10, where ‘12V supply’ is the positive contact and the ‘12V ground’ is the negative contact of a 12V battery pack.

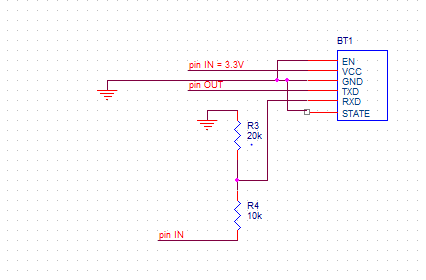


Figure : Circuit used for Bluetooth Module. Each Arduino has one.

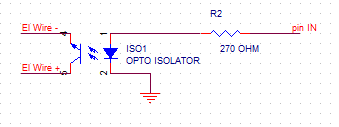


Figure : The Arduino tells when the switch (opto-isolator) turns on which then turns on the AC glowing EL Wire.

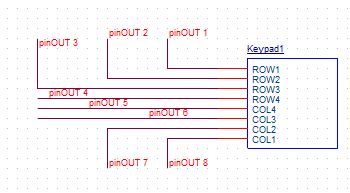


Figure : The planned connection between the keypad membrane to the NFC Arduino.

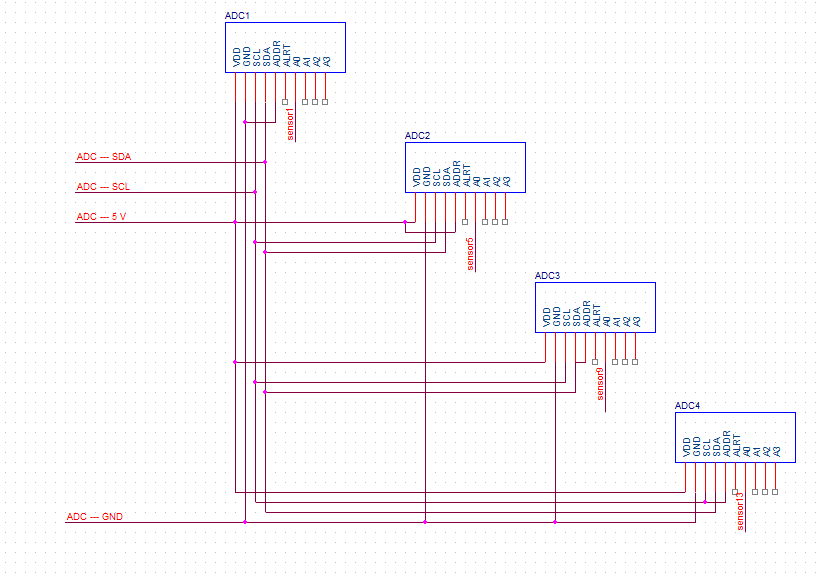


Figure : The wiring of the several ADCs needed for the System to one Arduino, the Laser Arduino.

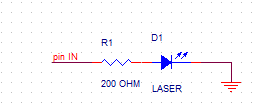


Figure : The final circuitry decided on to ensure the long life of the laser.

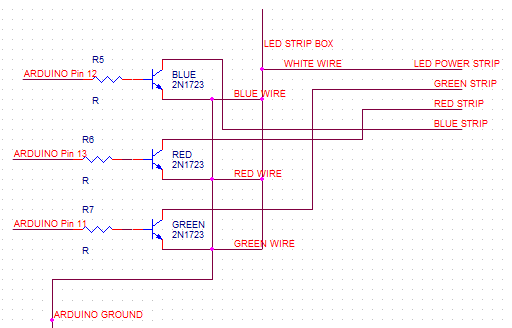


Figure : The circuit to ensure the programmatic use of the LED light strip where an analog signal can be used.

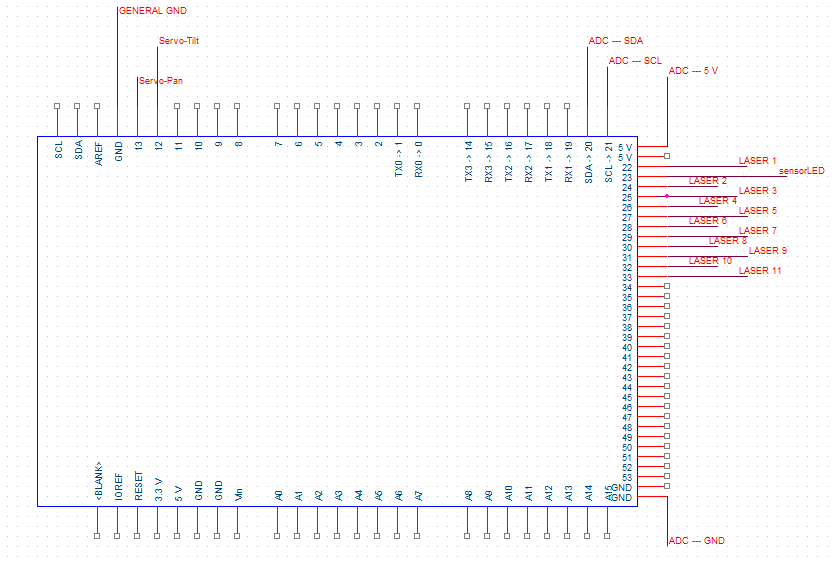


Figure : The layout of the pins and what uses them on the Laser Arduino.

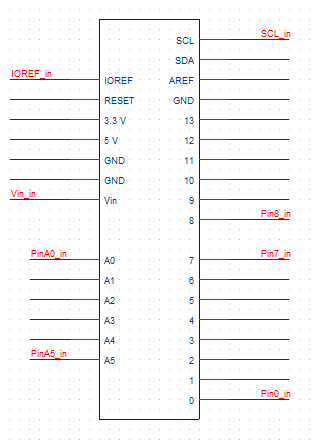


Figure : The pins the LoL shield uses. NOTE: All pins are used if used with an Arduino Uno.

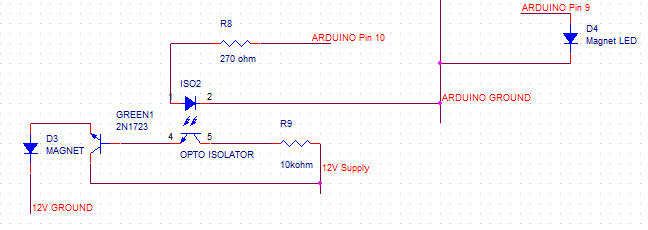


Figure : Circuit for both the door Magnet and the magnet arming LED.

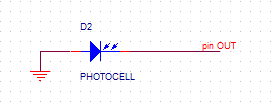


Figure : Circuit used for the solar cells when they are used as sensors. NOTE: ‘pin out’ is a wire which is then connected to an ADN channel which is then converted to a digital number for processing by the Laser Arduino.

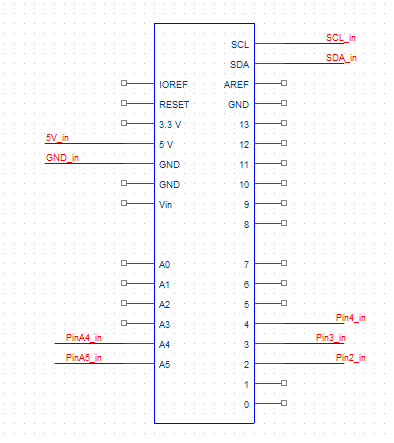


Figure : Schematic on which pins are used on the NFC (PN532) shield.

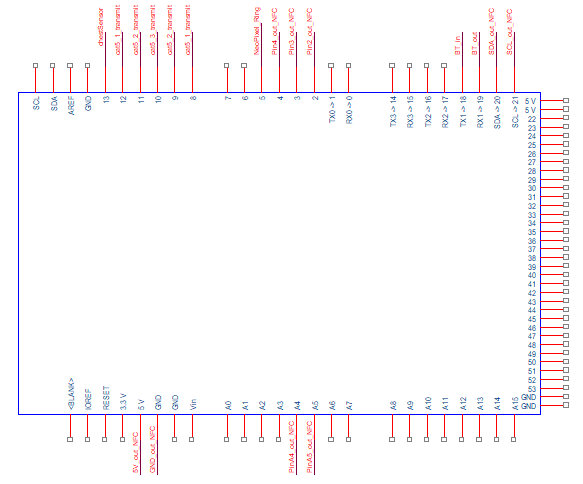


Figure :Schematic on which pins are used on the Arduino that controls the NFC shield.

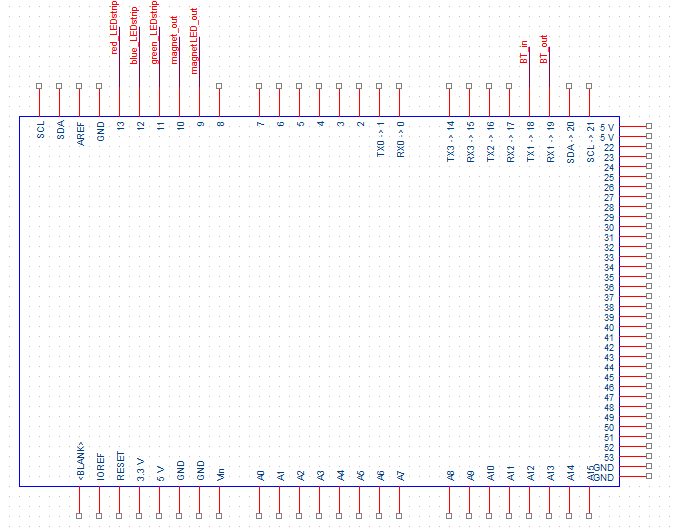


Figure : Schematic that shows what pins are used for the magnet and LED strip lighting.

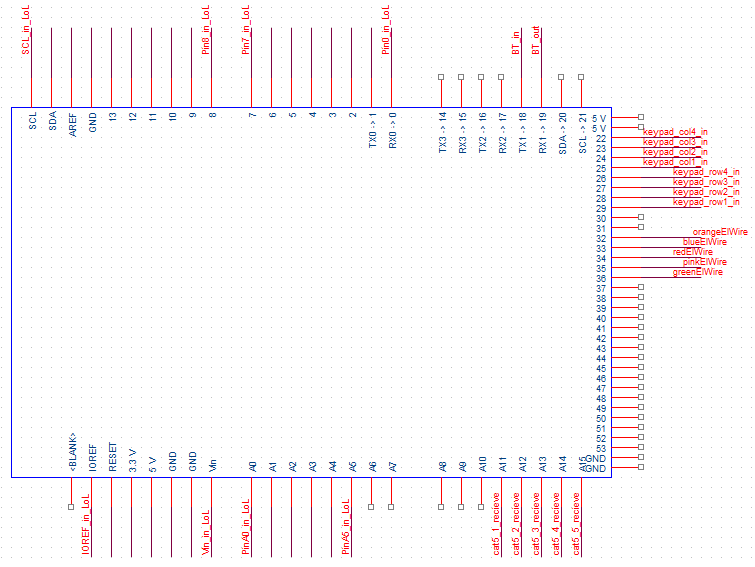


Figure : Schematic that shows what pins are used on the Arduino that controls the LoL shield.

## Devices and Their Placement in Arduino Files

Due to programming issues in the Arduino language, we include all necessary functions directly into the Arduino program code or else the functions/methods do not behave as expected. For explicit model numbers and other necessary information to buy the required parts, see the Budget section.

The Bluetooth module (BT module) we used in the System was an HC05 module where the module can either be in slave mode or it can be in master mode, and it can be either be in AT or non-AT state. In this System, we used AT state to configure the BT module for reliable communication between devices and we used the non-AT state as the communication state since communication between devices is not possible in AT state. The configuration details to get into the various states and to setup the BT module is mentioned in the below sub sections.

Bluetooth Module Setup

For the BT module Setup on the Arduino, we first had to start the Bluetooth communication and since we are using pins Rx1 and Tx1 on the Arduino we use the method ‘Serial1.begin(<baud rate>) in the Arduino program code and then place the module into AT state which is done by setting the ENABLE pin to 3.3V and then turning on the module. We then used the following configuration values:

AT mode (configuration/Admin):

baud rate : 38400

mode : slave

UART : 9600 (?)

non-AT mode (communication mode):

baud rate: 9600

There are also custom functions located in the files to manage getting useful data from the Raspberry Pi and another to send useful data back to the Raspberry Pi.

We use glowing El wire as the glowing wire in this System. We got one large coil from the Adafruit website and several smaller lengths from off the Amazon website. I then used the schematic shown in Figure 3 to wire one connection between the inverter/power supply to the wire and left the other connection between the inverter/power supply as a regular wired connection without any additional circuitry.

The glowing wires are all managed by the LoL Shield Arduino since I prefer all devices getting managed by the same Arduino whenever possible.

The keypad membrane is managed by the NFC Arduino because the NFC Arduino is closer to where the keypad was placed on the enclosure, and also has more memory free for use. The keypad sends data through digital pins back to the host Arduino, but we need to include and create a keypad object in order for the keypad to be easily used.

The ADCs (Analog-to-Digital Converter) are wired as shown in Figure 5 in order to enable one Arduino to handle more than one ADC. I got this wiring diagram off the Adafruit learning ADC tutorial. As you can see, it takes a SDA, SCL, 5V power, and a ground pin in order to use the ADCs. The ADCs allow us to take an analog value from our sensors and convert it into a useful digital value which we can check if it is valid on the Arduino. All the ADCs are connected to the Laser Arduino in order to provide a maximum value of 16 sensors we can manage without programmatically changing which sensor can get read from a specific ADC channel.

The circuit shown in Figure 6 is the circuit for each individual laser that is not in an enclosure with other hardware. There was one custom function where a laser can blink on and off, and this function and all other laser-related functionality can be found in the Laser Arduino.

The LED strip in the System is used for room lighting and all functionality is located within the Magnet Arduino, since the Magnet Arduino has extra space for electronics and has the LED strip connected to it.

The schematic shown in Figure 8 describes which pins on the Laser Arduino are connected to which device. Notice how the majority of the devices connected to the Laser Arduino are either lasers or sensors.

The schematic of the LoL shield which can be seen in Figure 9 is included for completeness of documentation. The LoL shield uses all pins that it can connect to and therefore there is no custom wiring done.

The wiring for the magnet was created so the 300mA that is required for the magnet will not touch any circuitry with the Arduino, and therefore all possible damage to the Magnet Arduino will be reduced.

The solar cell that can be seen in Figure 11has its negative contact connected to the Laser Arduino’s ground pin and its positive contact is connected to a channel on the ADC devices connected to the Laser Arduino. The analog value is then converted into a digital value for evaluation and sent from the ADC devices to the Laser Arduino. The management for the sensors involve an ADC object which can be found at the top of the Laser Arduino program (.ino) file, and the retrieving of the sensor data can be seen in the ‘loop’ function/method.

## Logic behind the Devices: Coding

This section describes the possibly confusing coding done within the System that may be difficult for a new System administrator/modifier to understand. As you may see later in this section, there are parts to the System that are not currently modular but could become modular in the future and there are parts to the System that are currently modular but could become non-modular in the future due to System administrator’s/modifier’s opinions.

### NFC Card and El Wire

The most complex part within the System in terms of Arduino programming code, are the code that manage the NFC cards in the NFC Arduino and the code that manages the glowing El wire which is located in the LoL Shield Arduino. Both of these were primarily created to be modular and therefore use function pointers. In both instances, we create an object that has the appropriate functions within and has an array of said functions. We then create an array of events, and then trigger the function within the object that corresponds with the observed event.

A good example of this setup is the setup for the NFC cards. For the NFC cards we have two important components, where the first important component was a major array which contains the UID (unique identification number) of all acceptable NFC cards and the second important component is the NFC card manager object which contains all the functionality when an acceptable NFC card is tapped. We then go through the first array when we observe an NFC card is tapped, and then we trigger the associated function from within the NFC card manager object. This works because the functions within the NFC card manager object are arranged into an array structure, where the first function in the NFC card manager object corresponds to the first UID in the NFC card array. This setup also requires the exact number of cards (and therefore functions in the NFC card manager) to be known and modified within the file.

### Bluetooth Communication

The code for the Bluetooth communication may be a little confusing since we used Arduino Megas for this System. There is an addition in the Arduino library to ensure communication over Tx and Rx pins are easier to manage in modern times in comparison to several years ago. This is important because we used the Serial1 object within our System where it explicitly deals with communication with Tx1 and Rx1 pins. If Tx2 and Rx2 or Tx3 and Rx3 combinations are used, then change all Serial1 objects to either Serial2 or Serial3 consecutively. The software for the Bluetooth module will also depend on the Arduino used so please do some research on how to use Bluetooth with your unique device if you are not using Arduino Megas.

## Maintenance for Adding/Replacing Devices

For adding or removing device to the System, please see the Circuit Schematics section to see what Arduino pins are available for devices and/or what pins to connect the replacement part to.

### Replacing Parts

Please refer to Devices and Their Placement in Arduino Files in order to modify any necessary code for the new replacement part, read the Circuit Schematics section to see what pins to connect the replacement part to, and the Testing section to see where the necessary testing code is located to ensure correct behaviour of the device.

### Adding Parts

Please see the Circuit Schematics section to see what Arduino pins are available for devices and ensure the necessary pins needed for the new device(s) are available and the new device(s) are compatible with the Arduino. In addition add the necessary code into the Arduino program (.ino) file, add the necessary testing code into the file mentioned in the Testing section of this document, update the Circuit Schematics section, and follow all the coding rules mentioned in the Coding Practices section.

## Testing

At the beginning of the project all testing code was implemented with the production code, but due to crowding of code and the decrease in readability of Arduino program code all testing code is now present in separate files from the production code. All testing code can be found in the ‘TestingCode’ directory at the creation of this document.

The testing code for the Bluetooth communication for Arduinos can be found in the BT\_Testing.ino file, and the testing code for both the NFC Arduino and the LoL Shield Arduino can be found in the cubeMainArduinoTesting.ino file.

The testing of the NFC Arduino can be completed by opening the cubeMainArduinoTesting.ino file, un-comment all lines in the ‘meant for NFC Arduino’ section and comment all other lines present in the setup and loop functions, and then compiling the code on the NFC Arduino.

The testing for the LoL Shield Arduino can be completed by following the same steps as mentioned for the NFC Arduino but replacing all ‘NFC Arduino’ with ‘LoL Shield Arduino’.

# Budget

The maximum amount of money to be used on materials of this project was projected to be $1000.00 CAD and the projection of costs was done by Oles Shnurovskyy who had the primary idea of this project and manager of the project. This section shows what the end budget for this System, along with projected prices for borrowed elements of the project, and with an estimated total for parts within the System. Notice that there are no estimations on work hours since all workers on the System are beginners and so a large amount of work hours were spent on researching various devices involved.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Budget as of September 20, 2017 | | | | |
| **Hardware** | | | | |
| part | #ofpart | priceofPart | total | who bought |
| keypad membrane 4x4 | 1 | 4.98 | 4.98 | oles |
| led strip | 1 | 9.99 | 9.99 | oles |
| magnet | 1 | 29.9 | 29.9 | oles |
| 6pack 22AWG wire | 2 | 20.13 | 40.26 | oles |
| cat5 cable | 4 | 1.99 | 7.96 | oles |
| el wire (5 pack) | 1 | 19.99 | 19.99 | oles |
| el wire (1 pack, orange) | 1 |  | 0 | oles |
| 3D printer filament (1kg) | 2 | 25 | 50 | oles |
| cube materials |  |  | 0 | oles |
| cat5e connectors |  |  | 0 |  |
| headlight/cube LED | 2 |  | 0 | oles |
| sw90 servo motor (5 pack) | 1 | 18.99 | 18.99 | oles |
| adc - adafruit | 4 | 9.95 | 39.8 | oles |
| bluetooth modules | 4 | 14.98 | 59.92 | oles |
| nfc reader/writer | 1 | 39.99 | 39.99 | chris |
| led resistors | 5 |  | 0 |  |
| optoisolator | 5 | 0.414 | 2.07 | chris |
| 5W laser | 17 | 5.36 | 91.12 | chris |
| solar cell sensor | 20 | 1.966 | 39.32 | chris |
| 5W laser | 1 | 5.99 | 5.99 | chris |
| TRANSISTOR NPN 60V 3A TO-92 | 4 | 1.349 | 5.396 | chris |
| neopixel led ring | 1 | 9.5 | 9.5 | chris |
| PanTilt motor | 1 | 18.95 | 18.95 | chris |
| connector strip - 50pin | 1 | 11.73 | 11.73 | chris |
| 200ohm resistor | 17 | 0.0656 | 1.1152 | chris |
| black, red, violet, orange wire | 4 | 2.95 | 11.8 | chris |
| RES 2K OHM 1/2W 5% AXIAL | 10 | 0.103 | 1.03 | chris |
| 1K resistor 1/4W | 6 | 0.00679 | 0.04074 | chris |
| 10k resistor 2W | 6 | 0.336 | 2.016 | chris |
| prototype breadboard | 1 | 1.64 | 1.64 | chris |
| terminal block | 2 | 0.589 | 1.178 | chris |
| 4x AA battery holder | 2 | 2.6 | 5.2 | chris |
| 9V battery snap 'I' type | 2 | 0.36 | 0.72 | chris |
| super bright yellow LEDs | 20 | 0.35 | 7 | chris |
|  |  |  |  |  |
| **miscellanous parts we didn’t use but bought** | | | | |
| led sequins | 1 (pack of 5) | 3.95 | 3.95 | chris |
| 9V battery clip (arduino input) | 3 | 2.95 | 8.85 | chris |
| grove electromagnet | 1 | 12.19 | 12.19 | chris |
| relay (500mA) | 17 | 1.1556 | 19.6452 | chris |
|  |  |  |  |  |
| **Parts borrowed and estimated cost** | | | | |
| part | #of part | pricePerPart | total |  |
| arduino Mega 2560 | 4 | 48.56 | 194.24 | legit arduinos |
| arduino programming cable | 4 | 5.83 | 23.32 |  |
| lol shield | 1 | 5.66 | 5.66 | obsolete, found on ebay |
| breadboard (3 pack) | 1 | 13.86 | 13.86 | bought - chris |
| male/male jumper wires | 2 | 3.95 | 7.9 | bought - chris |
| male/female jumper wires | 1 | 3.95 | 3.95 | bought - chris |
| use of 3D printer (printed parts) | - | 0.1 | - | price from UofR printing (per gram) |
| laser ball mounts | 17 | 12 | 20.4 |  |
| mirror ball mounts | 17 | 12 | 20.4 |  |
| cube texture |  |  | 0 |  |
| raspberry pi 2 stuff | 1 |  | 0 | bought- oles |
| raspberry pi 2 bluetooth dongle | 1 |  | 0 |  |
| raspberry pi 2 wifi dongle (do we use it?) | 1 |  | 0 |  |
| 12VDC -800mA wall charger | 1 | 4 | 4 | price off ebay |
| anker power bank 20000mAh | 1 | 44.99 | 44.99 | bought-chris |
|  |  |  |  |  |
| **Totals** | |  |  |  |
| oles total | 281.79 |  |  |  |
| chris total | 255.80594 |  |  |  |
| oles&chris total | 537.59594 |  |  |  |
| parts total | 537.59594 |  |  |  |
| all parts accounted for ? | YES |  |  |  |
|  |  |  |  |  |
| miscellaneous total | 44.6352 |  |  |  |
| borrowed parts total | 338.72 |  |  |  |
| borrowed & parts total | 876.31594 |  |  |  |

# Conclusion

Any maintenance for Escape Room 1 should be do-able with reference to this document, and fair amount of information should be given to reproduce this project at any given time with respect to devices used and functionality of devices. Feel free to modify the length of wire used and enclosure for devices used.

# References

L. (2012, December 30). Adafruit PN532 RFID/NFC Breakout and Shield. Retrieved October 01, 2017, from https://learn.adafruit.com/adafruit-pn532-rfid-nfc/mifare